CMPSC265 Data Structures and Algorithms
Homework 10

Due: Dec 1st, 2019, 11:59pm

Learning Objectives

- Graph
- Depth-First-Search and Breadth-First-Search

Deliverables:

- The modified java file Graph.java for Problem 1, Problem 2 and Problem 3.
-

Instructions:

- Please first download the Java source file Graph.java, modify the codes based upon requirements, and submit the updated source file. Please do NOT change the provided part of the codes, but you may add additional methods if needed.
- Please add the required *credit comments* and *comments* to each Java program you submit.
- **Please TRY YOUR BEST before seeking help from any online resources or tutors or other people. If you have asked for any kind of help, you need also submit your own draft before getting the help, and write a report of how you got helped and what you have learned from the help.**

Problem Specifications:

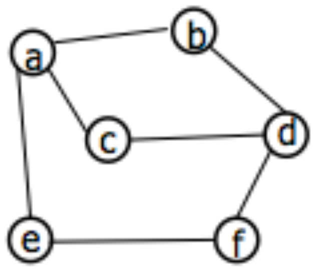
Problem 1: Detecting Connected Component on an Undirected Graph (32')

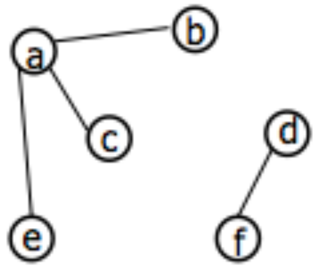
The attached GraphProcess.java file contains the definition for the Vertex Class as well as for the Graph class.

Please finish the implementation of the method of *detectConnectedComponent()* which will find out all the connected components in an undirected graph.

It is known that a connected component in a graph is a sub-graph of the original graph in which there is at least one path between every two vertices (nodes).

For example, the figure on the left contains only one connected component, which is the graph itself. However, the figure on the right contains two connected components: Nodes set {a, b, c, d, e} forms one connected component, and the Nodes set {d, f} forms another one.





The method will return an ArrayList of ArrayList, which contains the value of each vertex in a connected component.

You can follow the idea of either Depth-First-Search (DFS) or Breadth-First-Search(BFS) to do it.

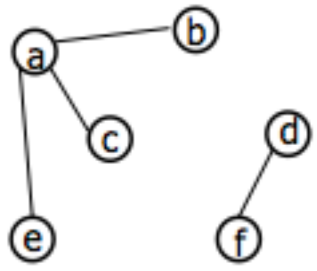
Apply DFS or BFS on a Graph, all the nodes on a connected component can be visited.

<u>Outputs:</u>

```
$ java GraphProcess
There are two connected components in this undirected
graph.
They are:
Connected Component 1: [a, b, c, e]
Connected Component 2: [d, f]
```

Problem 2: ***Finding connectivity between two nodes (32')***
<u>Description:</u>
Please finish definition of the *findPath()* method within the GraphProcess class, so that given two vertices, it can detect whether there exists a path them, and therefore they are connected.

For example, on the following figure, vertex *a* and *b* are connected, but vertex a and f are not.



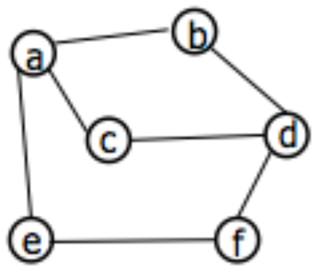
You can apply the idea of either DFS or BFS to implement this. If two vertices are connected, staring DFS or BFS on either one node can reach the other. **Please do not directly use the returned results from Problem 1.**

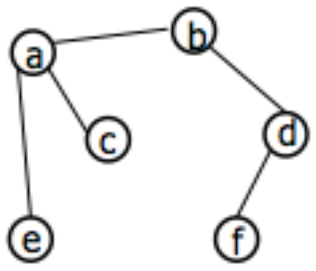
__**Outputs:**__
```
$java GraphProcess
Please enter vertex 1: a
Please enter vertex 2: b
These two vertices are connected.
```

Problem 3: Detecting Cycles on an Undirected Graph (34')
__**Description:**__
Please finish the implementation of method *hasCycle()* which can detect whether exists any cycle in an undirected graph.

You can use a recursive DFS to implement this. The basic algorithm is as: **Start DFS, return true if you see a visited node that is not your parent (calling node). If the search is done and it dose not return true, return false.**

For example, the left graph contains a cycle, but the right one does not.





Outputs:

```
$java GraphProcess
This graph contains cycles.
```